Package 'BrowserViz'

June 4, 2026

Type Package

Title BrowserViz: interactive R/browser graphics using websockets and JSON

Version 2.35.1

Date 2023-09-12

Author Paul Shannon

Maintainer Arkadiusz Gladki <gladki.arkadiusz@gmail.com>

Depends R (>= 3.5.0), jsonlite (>= 1.5), httpuv(>= 1.5.0)

Imports methods, BiocGenerics

Suggests RUnit, BiocStyle, knitr, rmarkdown

Description Interactvive graphics in a web browser from R, using websockets and JSON.

License GPL-2

URL https://gladkia.github.io/BrowserViz/

BugReports https://github.com/gladkia/BrowserViz/issues

LazyLoad yes

biocViews Visualization, ThirdPartyClient

NeedsCompilation no

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

git_url https://git.bioconductor.org/packages/BrowserViz

git_branch devel

git_last_commit fce94bb

git_last_commit_date 2026-05-01

Repository Bioconductor 3.24

Date/Publication 2026-06-04

Contents

`addRMessageHandler` *Supply the name of a function to call, identified by its key*

Description

Supply the name of a function to call, identified by its key

Usage

```
addRMessageHandler(key, functionName)
```

Arguments

`key`	A character string
`functionName`	A character string

browserResponseReady,BrowserViz-method
browserResponseReady

Description

browserResponseReady

Usage

```
## S4 method for signature 'BrowserViz'
browserResponseReady(obj)
```

Arguments

obj	An object of class BrowserViz

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  browserResponseReady(bvApp)
  closeWebSocket(bvApp)
  }
```

BrowserViz constructor
Constructor for BrowserViz

Description

This constructor function:

- creates the BrowserViz object
- initializes the httpuv web server
- prepares that web server to additionally handle websocket traffic
- loads a "browserFile" - an html/javascript/css web page to communicate with in your web browser
- opens websocket communication between your R session and your browser
- installs an optional "httpQueryProcessingFunction" to handle http (non-websocket) requests.

Usage

```
BrowserViz(
  host = "localhost",
  portRange = 10000:10100,
  title = "BrowserViz",
  browserFile,
  quiet = TRUE,
  httpQueryProcessingFunction = NULL
)
```

Arguments

host	character The constructor will open an http/websocket port here for web browsers to connect to. localhost by default
portRange	The constructor looks for a free websocket port in this range. 10000:10100 by default
title	Used for the web browser window, "igvR" by default
browserFile	The full path to the bundled html, js and libraries, and css which constitute the browser app
quiet	A logical variable controlling verbosity during execution
httpQueryProcessingFunction	a function, default NULL, provides subclasses with the opportunity to execute code on the http server created here.

Value

An object of the BrowserViZ class

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  data <- list(lowercase=letters, uppercase=LETTERS)
  json.returned <- roundTripTest(bvApp, data)
  data.returned <- fromJSON(json.returned)
  stopifnot(identical(data, data.returned))
  html <- sprintf("<h3>round trip of json-encoded data, %d chars</h3>",
                  nchar(json.returned))
  displayHTMLInDiv(bvApp, html, "bvDemoDiv")
  closeWebSocket(bvApp)
  }
```

`BrowserViz-class` *BrowserViz: a base class providing simple, extensible message passing between your R session and your web browser, for interactive data visualization.*

Description

Many of the best interactive graphics capabilities available today are written in Javascript and run in a web browser. BrowserViz makes these capabilities available in R, using websockets for message passing back and forth between R and the browser. This class connects your R session to your web browser via websockets, using the R httupv library, which in turn uses the Rook webserver.

BrowserViz is a concrete base class, in that instances can be constructed and run - which we do for testing. The primary use of this BrowserViz is to be subclassed: to facilitate the creation of new browser-based, R-connected interactive graphics capabilities embodied in R packages, written by programmers with some skill in both R and Javascript. Two examples of this can be found in these Bioconductor packages https://bioconductor.org/packages/devel/bioc/html/igvR.html and https://bioconductor.org/packages/devel/bioc/html/RCyjs.html.

Slots

`uri` The http location at which this modest webserver runs

`port` An integer port number for the http connection

`websocketConnection` An environment managed by the httpuv library on our behalf

`quiet` Logical varaible controlling verbosity during execution

See Also

`BrowserViz`

An S4 class to create and manage a modest webserver for websocket message passing between R and Javascript running in your web browser

`closeWebSocket,BrowserViz-method` *Close the websocket connection - between your R session and your web browser.*

Description

Close the websocket connection - between your R session and your web browser.

Usage

```
## S4 method for signature 'BrowserViz'
closeWebSocket(obj)
```

Arguments

`obj`	An object of class BrowserViz

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  show(bvApp)
  closeWebSocket(bvApp)
  }
```

`dispatchMessage` *Route the message coming in from the browser to the appropriate R function.*

Description

Route the message coming in from the browser to the appropriate R function.

Usage

```
dispatchMessage(ws, msg, quiet)
```

Arguments

`ws`	a websocket connectin
`msg`	the JSON-encoded message from the browser
`quiet`	logical TRUE or FALSE

`displayHTMLInDiv,BrowserViz-method`
Ask the browser to display html markup in the specified div

Description

Ask the browser to display html markup in the specified div

Usage

```
## S4 method for signature 'BrowserViz'
displayHTMLInDiv(obj, htmlText, div.id)
```

Arguments

`obj`	An object of class BrowserViz
`htmlText`	A character string with HTML markup
`div.id`	A character string

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  data <- list(lowercase=letters, uppercase=LETTERS)
  json.returned <- roundTripTest(bvApp, data)
  html <- sprintf("<h3>round trip of json-encoded data, %d chars</h3>",
                  nchar(json.returned))
  displayHTMLInDiv(bvApp, html, "bvDemoDiv")
  closeWebSocket(bvApp)
  }
```

---

fromJSON *Transform JSON string into a native R object*

Description

Transform JSON string into a native R object

Usage

```
fromJSON(...)
```

Arguments

...	Extra arguments passed to this function

Value

a native R data structure

Examples

```
fromJSON(toJSON(data.frame(a=8:10, b=LETTERS[8:10], stringsAsFactors=FALSE)))
```

---

getBrowserInfo,BrowserViz-method *Retrieve basic attributes of the attached web browser.*

Description

Retrieve basic attributes of the attached web browser.

Usage

```
## S4 method for signature 'BrowserViz'
getBrowserInfo(obj)
```

Arguments

obj	An object of class BrowserViz

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  getBrowserInfo(bvApp)
  closeWebSocket(bvApp)
  }
```

---

getBrowserResponse,BrowserViz-method

Retrieve the response sent by the browser

Description

Retrieve the response sent by the browser

Usage

```
## S4 method for signature 'BrowserViz'
getBrowserResponse(obj)
```

Arguments

obj	An object of class BrowserViz

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  data <- list(lowercase=letters, uppercase=LETTERS)
  json.returned <- roundTripTest(bvApp, data)
  data.returned <- fromJSON(json.returned)
  stopifnot(identical(data, data.returned))
  closeWebSocket(bvApp)
  }
```

getBrowserWindowSize,BrowserViz-method

Supply the name of a function to call, identified by its key

Description

Supply the name of a function to call, identified by its key

Usage

```
## S4 method for signature 'BrowserViz'
getBrowserWindowSize(obj)
```

Arguments

obj	An object of class BrowserViz

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  getBrowserWindowSize(bvApp)
  closeWebSocket(bvApp)
  }
```

getBrowserWindowTitle,BrowserViz-method

Supply the name of a function to call, identified by its key

Description

Supply the name of a function to call, identified by its key

Usage

```
## S4 method for signature 'BrowserViz'
getBrowserWindowTitle(obj)
```

Arguments

obj	An object of class BrowserViz

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  getBrowserWindowTitle(bvApp)
  closeWebSocket(bvApp)
  }
```

---

handleResponse *handleResponse*

Description

handleResponse

Usage

```
handleResponse(ws, msg)
```

Arguments

ws	websocket connection
msg	the JSON-encoded character string returned by the browser

---

port,BrowserViz-method *Get the port number*

Description

Get the port number

Usage

```
## S4 method for signature 'BrowserViz'
port(obj)
```

Arguments

obj	An object of class BrowserViz

Value

the port number use in the websocket connection, a numeric value.

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  port(bvApp)
  closeWebSocket(bvApp)
  }
```

---

ready,BrowserViz-method

Is the websocket connection to the browser ready for use?

Description

Is the websocket connection to the browser ready for use?

Usage

```
## S4 method for signature 'BrowserViz'
ready(obj)
```

Arguments

obj	An object of class BrowserViz

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  ready(bvApp)
  closeWebSocket(bvApp)
  }
```

---

roundTripTest,BrowserViz-method

Send data to the browser, ensure that it is returned accurately.

Description

Send data to the browser, ensure that it is returned accurately.

Usage

```
## S4 method for signature 'BrowserViz'
roundTripTest(obj, ...)
```

Arguments

obj	An object of class BrowserViz
...	other arguments

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  data <- list(lowercase=letters, uppercase=LETTERS)
  json.returned <- roundTripTest(bvApp, data)
  data.returned <- fromJSON(json.returned)
  stopifnot(identical(data, data.returned))
  closeWebSocket(bvApp)
  }
```

---

send,BrowserViz-method

Send the specified message to the browser

Description

Send the specified message to the browser

Usage

```
## S4 method for signature 'BrowserViz'
send(obj, msg)
```

Arguments

obj	An object of class BrowserViz
msg	A list with four fields: cmd, status, callback, payload e.g. list(cmd="someCommand", status="request", callback="someFunction", payload="someData")

---

setBrowserWindowTitle,BrowserViz-method

Supply the name of a function to call, identified by its key

Description

Supply the name of a function to call, identified by its key

Usage

```
## S4 method for signature 'BrowserViz'
setBrowserWindowTitle(obj, newTitle)
```

Arguments

obj	An object of class BrowserViz
newTitle	A character string

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  getBrowserWindowTitle(bvApp)
  setBrowserWindowTitle(bvApp, "testBrowser")
  getBrowserWindowTitle(bvApp)
  closeWebSocket(bvApp)
  }
```

---

show,BrowserViz-method

Display the core attributes of the BrowserViz object to stdout

---

Description

Display the core attributes of the BrowserViz object to stdout

Usage

```
## S4 method for signature 'BrowserViz'
show(object)
```

Arguments

object	An object of class BrowserViz

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  show(bvApp)
  closeWebSocket(bvApp)
  }
```

toJSON *Transform an R data structure into JSON*

Description

The semantics of toJSON changed between RJSONIO and jsonlite: in the latter, scalars are promoted to arrays of length 1. rather than change our javascript code, and since such promotion – while sensible in the context of R – strikes me as gratuitous, I follow jeroen ooms suggestion, creating this wrapper

Usage

```
toJSON(..., auto_unbox = TRUE)
```

Arguments

...	Extra arguments passed to this function
auto_unbox	Logical

Value

a character string with the JSON representation of the R object

Examples

```
toJSON(data.frame(a=8:10, b=LETTERS[8:10], stringsAsFactors=FALSE))
```

wait,BrowserViz-method *Pause for the specified number of milliseconds*

Description

Pause for the specified number of milliseconds

Usage

```
## S4 method for signature 'BrowserViz'
wait(obj, msecs)
```

Arguments

obj	An object of class BrowserViz
msecs	Numeric

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  wait(bvApp, 100)
  closeWebSocket(bvApp)
  }
```

---

webBrowserAvailableForTesting

Is there a web browser available for testing?

Description

This package's unit tests require a web browser to connect to. our heuristic, though not bullet proof, is that one of three conditions must be met Supply the name of a function to call, identified by its key

Usage

```
webBrowserAvailableForTesting()
```

Value

Logical TRUE or FALSE

Examples

```
library(BrowserViz)
browserVizBrowserFile <- system.file(package="BrowserViz", "browserCode", "dist", "bvDemoApp.html")
if(BrowserViz::webBrowserAvailableForTesting()){
  bvApp <- BrowserViz(browserFile=browserVizBrowserFile, quiet=TRUE)
  data <- list(lowercase=letters, uppercase=LETTERS)
  json.returned <- roundTripTest(bvApp, data)
  html <- sprintf("<h3>round trip of json-encoded data, %d chars</h3>",
                  nchar(json.returned))
  displayHTMLInDiv(bvApp, html, "bvDemoDiv")
  closeWebSocket(bvApp)
  }
```

Index